# Niryo Studio User Manual

Niryo Studio is a graphical HMI. It allows a fast and direct control of Ned with an external computer.

Its purpose is to provide users with a complete and simple interface for Ned motion, programming environments and current status of Ned.

Through Niryo Studio application, users can:

- Manage and set Ned parameters
- Control Ned motion
- Program Ned using Blockly
- Execute a program
- Manage, set, control and program Ned tools
- Manage, set, control and program Ned addons

Users must read this manual in order to fully understand all Ned and Niryo Studio functionalities.

**Note**

If you encounter software problems or if you want to access latest updates for your Ned, please give an internet access to your robot (Ethernet or Wi-Fi) to be able to update it through NiryoStudio. Instructions here (index.html#ned-software-update)

## Changelog

### v3.2.2

- German translations updated

### v3.2.1

- Fix popups information / URLs relatives to robot update
- Documentation improvements

## September release - New features batch

### Features

#### Direct control

- Jog Pose.

  Control the pose of the robot in position and orientation. The arrows show the direction of the translation and rotation command to apply to the robot’s pose. You can adjust the command speed.

- Jog Joints.

  Control the joints of the robot in orientation. The arrows show the direction of the rotation command to apply to a joint. You can adjust the command speed.

#### State and 3D View

- Frames visualization.

  Visualize the TCP and the robot base frames. You can select which frames you want to display.

#### Programming

- **Try Except** block in *Logic* group.

  The function will try to run what is inside of it. A number of attempts can be set to repeat the operation if the robot failed to do it. You can set the function to continue or stop the program if the attempts do not succeed.

- **Linear** and **Try Linear in** *Move Pose* block, in *Movement* group.

  **Linear** function makes the robot move linearly. **Try Linear** function makes the robot move linearly but if no linear trajectory can be computed, it will do a *Standard* movement.

- **Linear** and **Try Linear** in *Shift pose* block, in *Movement* group.

  **Linear** function makes the robot do a translation movement linearly. **Try linear** makes the robot do a translation movement linearly but if no linear trajectory can be computed, it will do a *Standard* movement.

- **Move Trajectory** function in *Movement* group.

This function builds a trajectory by giving it waypoints. The robot will try to reach way points in the given order. The *distance smoothing* parameter defines the distance from which the trajectory will be smoothed around each way point.

- **Set TCP** block in *Tool* group.

  This function changes the TCP (Tool Center Point) coordinates of the robot. TCP coordinates are used with *Pose*-type movements.

- **Activate/Deactivate TCP** block in *Tool* group.

  This function activates or deactivates TCP coordinates. When activated, TCP coordinates are defined by the user. When deactivated, the TCP coordinates are equal to the final point of the robot's arm.

### Settings

- Angles unit selection.

  Angles unit selection between degree and radian from the top bar.

- Setup TCP coordinates from *Robot's settings* view.

  Change TCP (Tool Center Point) coordinates. TCP coordinates are used as references for *Pose* movements. *Activate* TCP allows the modification of the TCP coordinates. *Scan* will check if a tool is attached to the arm and will update the TCP coordinates according to the connected tool. *Reset* will reset the TCP coordinates to the actual tool (0 if no tool). You can manually modify the TCP coordinates by writing your orientation and translation. Click *Apply* to apply the new coordinates.

### Saved positions

- Play position

  Move to the desired position from position list by clicking on the*Play* button.

### Vision set

- Camera's settings

  Brightness, saturation and contrast adjustments for the camera.

### Improvements

### Programming

- Graphical improvement of **save program** function in the *Blockly program* view.
- Graphical improvement of **Load program from robot** function in the *Blockly program* view.
- Remove **Clear workspace** in the *Blockly program* view.
- Graphical improvement of **Add position** function in the *Blockly program* view.

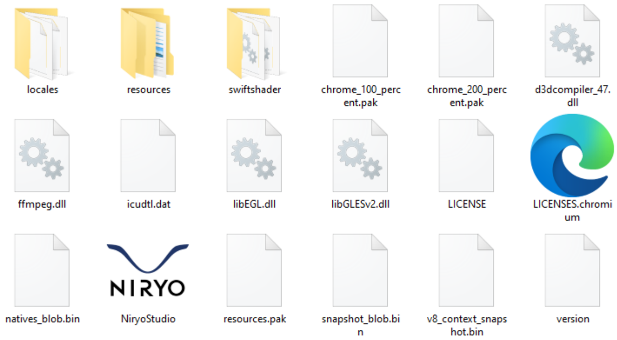
## Download Niryo Studio

You can download the archive containing Niryo Studio from Niryo Websitehere (https://niryo.com/en/download/). Select the version you need depending on your computer operating system.

## How to launch Niryo Studio

Once you have downloaded the .zip file you need to extract the archive. Then go into the new created folder and search for an executable named "NiryoStudio". Double click on the executable to start the application.

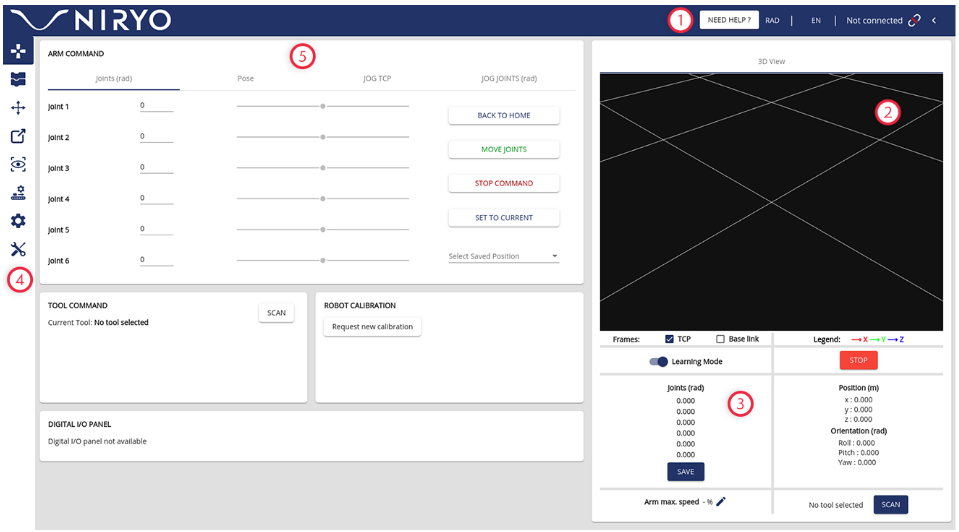
**You do not need to install anything on your computer.**



# Overview

## Operating interface

When the Niryo Studio launches, the window below appears.



1. The top toolbar allows you to change the angle units, the language, and connect to Ned
2. 3D Ned visualization or camera stream video

3. State section and mode selection
4. The left menu allows you to switch between the different sections of the application
5. The main window of the application

> **Note**
>
> If you have display problems with the positioning of certain elements. You can press ctrl and scroll wheel to zoom out the screen.

### Navigation interface

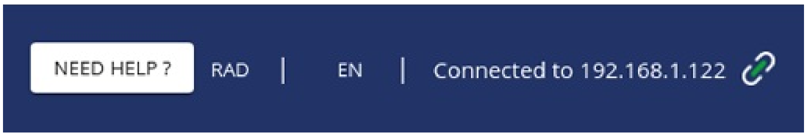
The listed icons from top to bottom on the left menu are:

- Direct command: allows direct motion control of Ned and tools
- Blockly Programming: create, edit and execute a program in Blockly.
- Saved positions
- Program settings: run and upload the robot's program
- Vision Set: set and program the Vision Set
- Conveyor Belt: scan and control the Conveyor Belt
- Settings: robot and software settings
- Logs and Status: displays robot, Niryo Studio logs and the hardware status

## Robot connection

There are three ways to connect Ned to Niryo Studio:

- Wired connection (Ethernet cable)
- Wireless connection (Wi-Fi):
  - Hotspot mode (the robot provides its own Wi-Fi network)
  - Connected mode (the robot is connected on your network)



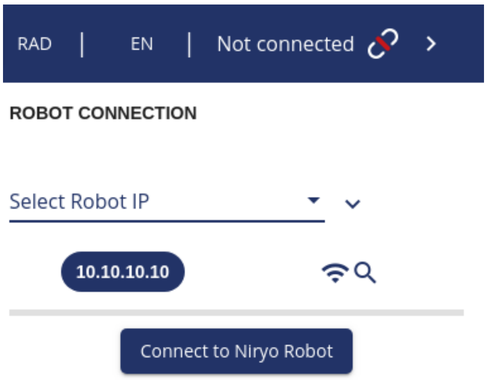
Connected to 192.168.1.122

### Using Ned in Hotspot mode

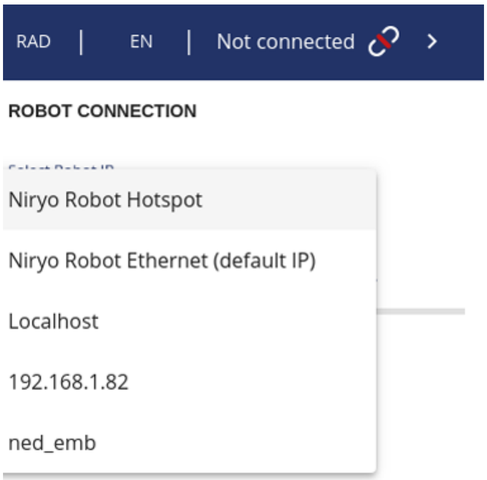
If the LED on the back panel is blue, that means you can connect to Ned via its own Wi-Fi. In this case Ned IP address is 10.10.10.10.

Ned has its own Wi-Fi network, which starts with “NiryoRobot” followed by a series of numbers and letters. To connect, follow the steps below:

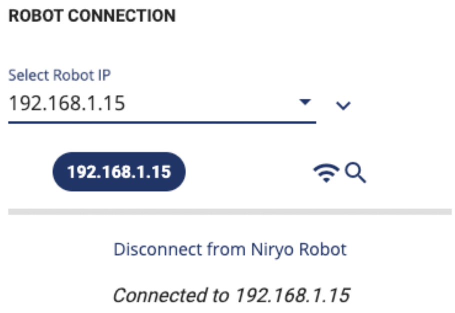
- Connect to the Wi-Fi access point using your laptop with the password **“niryorobot”**
- Click the arrow near “Not connected” to access Ned connection



- Select “Niryo Robot Hotspot” and connect



- To disconnect from Ned, open the connection panel again and click on “Disconnect”



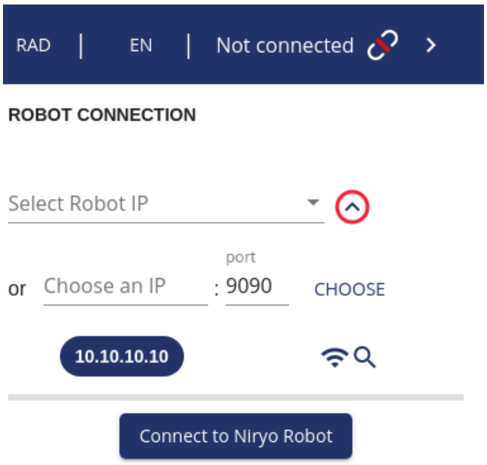
*Connected to 192.168.1.15*

## Using Ned on your Wi-Fi network

If Ned is connected to the Wi-Fi network (see the robot settings (index.html#ned-settings) section), the LED on the back panel is green.

To connect Ned to Niryo Studio, please follow the following steps:

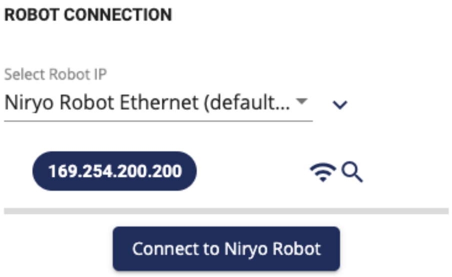
- Connect your computer to the same Wi-Fi network as Ned,
- If you do not know your robot IP address, open the connection panel and click on the "Wi-Fi + Search" button. This will search for connected robots inside the current Wi-Fi network. Choose the correct IP address and click "Connect Niryo Robot",
- If you gave a custom name to Ned, select it by its name,
- If you already know your Ned's IP address, click the arrow, enter your IP address and connect to the robot.



## Using Ned with an Ethernet cable

You can also use an Ethernet cable to physically connect Ned with your computer.

- Connect Ned to your local network (computer or switch) using the LAN port,
- Modify the selected IP address on the connection panel to "Niryo Robot Ethernet (default IP)",

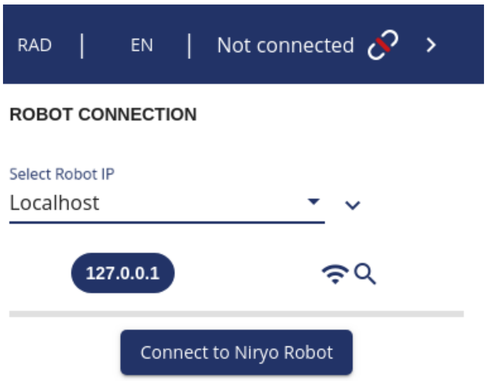


- Connect to Ned.

## Using Ned in simulation with Niryo Studio

Simulators such as Rviz (https://docs.niryo.com/dev/ros/source/simulation.html#control-with-ros) or Gazebo (https://docs.niryo.com/dev/ros/source/simulation.html#launch-simulation) must be launched on the same computer as the one running Niryo Studio.

- Click the arrow near "Not connected" to access the robot connection,



- Select “Localhost” and connect.

# Robots specificities

## Calibration

Ned needs to be calibrated in three cases:

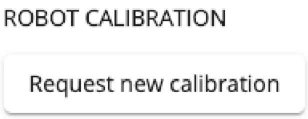
- At the start-up
- If the steppers motors lose their offset positions and need to re-calibrate
- If the 3D view and Ned physical movements are not similar

In most of the cases, Niryo Studio will alert you by showing “CALIBRATION NEEDED” on the top right.

**Note**

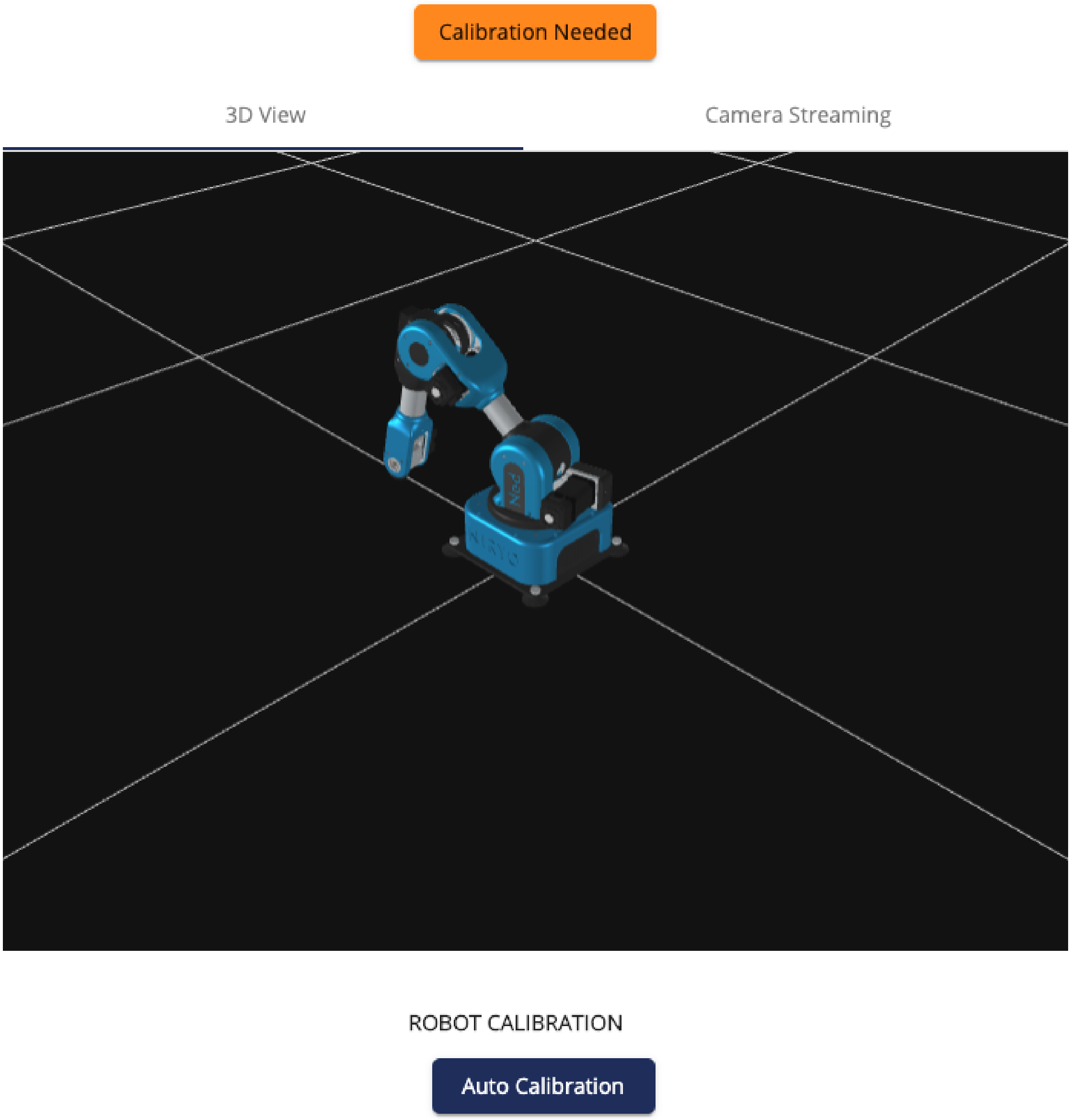
In the case of “CALIBRATION NEEDED”, you can’t perform any command on Ned without calibrating it.

To perform a calibration, click on the button “Request new calibration” in the “ROBOT CALIBRATION” section.



After clicking, a “Calibration Needed” warning will appear on top of the 3D view.

When the “Calibration Needed” warning appears, you can click on it to perform an auto calibration or you can click on the “Auto calibration” button in the “ROBOT CALIBRATION” section.



By choosing auto-calibration, Ned will execute an automatic sequence by moving its axis that needs to be calibrated until they reach their maximum position, so a correct offset can be applied to each motor. This sequence should take about 20-30 seconds. When the calibration is over, Ned will come back to a resting position.

**Caution**

Before launching the auto-calibration, make sure that there is no obstacle around Ned. You should put Ned in a proper position before performing the calibration. The position in the picture below is recommended. This will avoid damaging Ned, collision, and any hazard motion.

**Note**

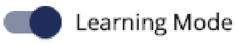
If the calibration is not successful, the “Calibration Needed” warning will still be displayed. Try the calibration again and if this new calibration does not succeed, it can mean that your robot has a mechanical/hardware issue.

**Hint**

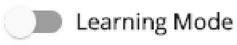
You can check if the calibration has succeeded by moving the robot and examine the robot motion in the 3D view.

## Learning mode

Niryo robots have two operating modes: the manual mode (Learning mode) and the automatic mode. The current mode can be determined visually in Niryo Studio, via the Learning Mode button position.



Learning Mode



Learning Mode

- **Manual mode/learning mode: when Ned is in learning mode, the toggle on Niryo Studio is on.**

  In this mode, the user can guide Ned by hand. In this case, there is no torque applied on the joints and motors are in relaxed mode. This mode is recommended when no program is running.

- **Automatic Mode (Learning Mode is deactivated): The Learning Mode does not work under Automatic Mode.**

  It is not possible to move Ned by hand. This mode is active when a program is running or when Ned is performing a motion.

**Caution**

Do not try to override the motor’s torque when learning mode is disabled: you could damage the robot and the motors. You can force to eliminate the risk or you can unplug the power supply ONLY in case of emergency (Ned blocked or application freezed)

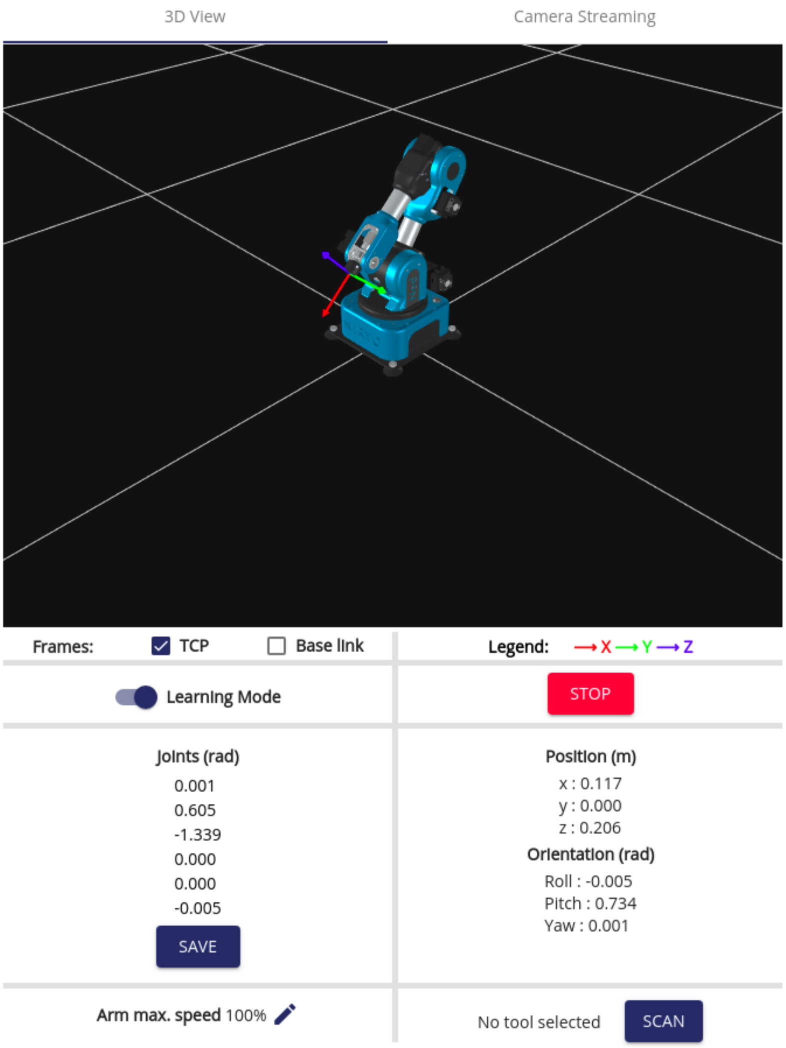
**Hint**

Activate the learning mode when the program is paused or stopped.

### Free motion (For Ned2)

For Ned2, the term “Learning mode” has been replaced by “Free motion”, but the functionality is identical. To use it, you now have to press and hold the “Free motion” button to activate it.

## State and 3D View



Frames: ☑ TCP □ Base link

Legend: → X → Y → Z

● Learning Mode

Joints (rad)
0.001
0.605
-1.339
0.000
0.000
-0.005

Position (m)
x : 0.117
y : 0.000
z : 0.206

Orientation (rad)
Roll : -0.005
Pitch : 0.734
Yaw : 0.001

Arm max. speed 100%

No tool selected

### 3D View

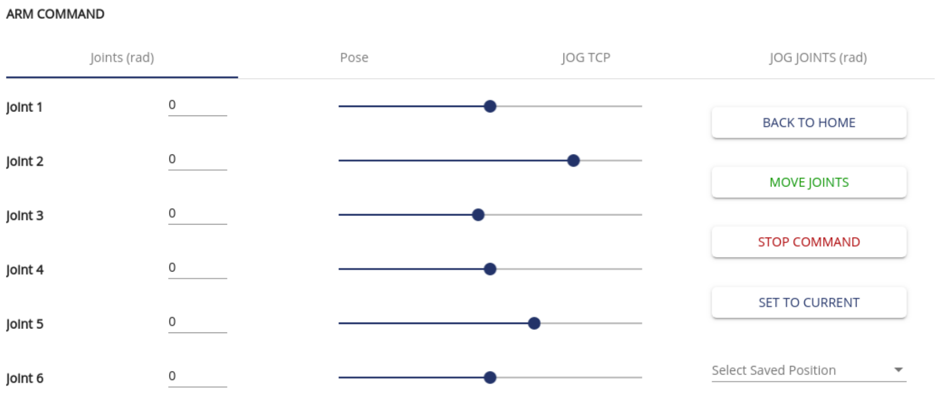
You can see here the 3D representation of the robot. You can display TCP (Tool Center Point) and Base link frames.

### States and commands

From here, you can:

- View the current joints angle.
- View the current TCP position and orientation.
- Activate or deactivate the Learning mode.
- Stop the current movement.
- Save the current position of the robot.
- Set the arm speed limit.

## Direct control



ARM COMMAND

Joint 1: 0
Joint 2: 0
Joint 3: 0
Joint 4: 0
Joint 5: 0
Joint 6: 0

### Arm command

On the ARM COMMAND section, you can move Ned's arm directly either by moving the robot joint individually or by translating / rotating Ned pose.

You will find 4 main subsections that this documention will cover below, but we will start with the main buttons available.

**Home Position**

Click the "BACK TO HOME" button in order to move Ned to its home position.

**Select Saved position**

If you already saved a position, you can select it via the "Select Saved Position" scrolling menu. Once selected, it will upload joints value if your are in joints view, or pose if you are in pose view.

Then you can click on "MOVE JOINTS" or "MOVE POSE" to move the robot according to the selected joints/pose.

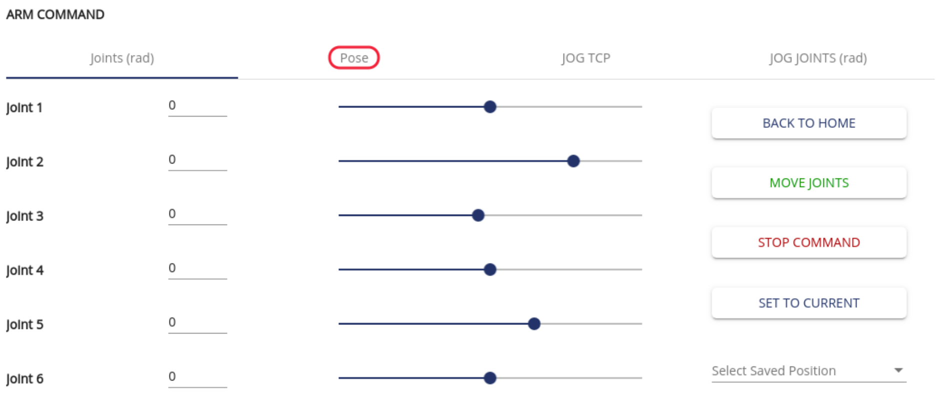
**Stop command**

The "STOP COMMAND" button stops Ned's motion.

**Joints**

This method allows you to control each joint individually from its minimum to its maximum value (see the Robots specificities (index.html#document-source/specification) section).

If a joint reaches its joint limit, it can't be driven any further.



| | |
|---|---|
| Joint 1 | 0 |
| Joint 2 | 0 |
| Joint 3 | 0 |
| Joint 4 | 0 |
| Joint 5 | 0 |
| Joint 6 | 0 |

By clicking on the "SET TO CURRENT" button, you will refresh the interface to the actual joints values.

To perform a move joints, you need to:

- Modify the values of the joints either by writing the desired joint value or by moving the cursor,
- Click on the "MOVE JOINTS" button to execute the move command.

Once the command is done (success or not), you will get a notification at the bottom of the screen.

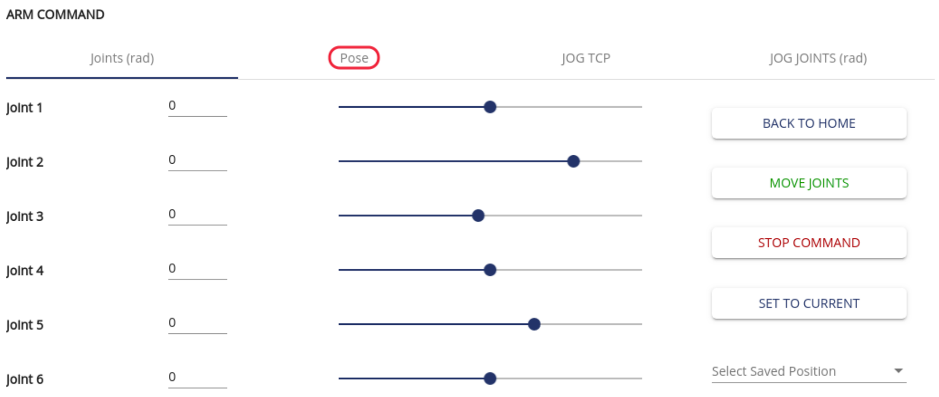
**Important**

At any time, you can cancel the current command execution by clicking on the "STOP COMMAND" button.

## Pose

This panel gives you the possibility to move the pose of Ned's TCP (Tool Center Point).

You can edit the values by clicking on them then use the "MOVE POSE" button.



| | |
|---|---|
| Joint 1 | 0 |
| Joint 2 | 0 |
| Joint 3 | 0 |
| Joint 4 | 0 |
| Joint 5 | 0 |
| Joint 6 | 0 |

## JOG TCP

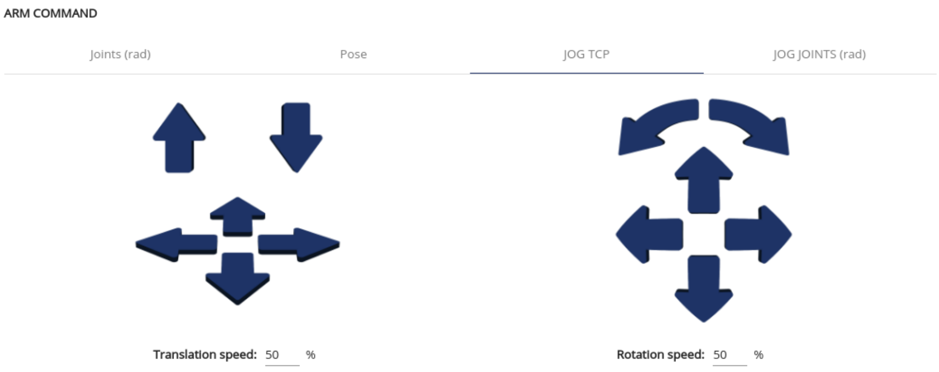
This panel will help you move Ned in a more visual way.

The TCP (Tool Center Point) of the robot is the center of the end effector.

- Holding a translational arrow will make the robot follow its direction,
- Holding a rotational arrow, will make the robot rotate around the TCP.

The robot will move until you release the button.

You can modify the speed of the jogs. A low value will give you more precision while a higher value will bring you faster to the position.



Translation speed: 50 %

Rotation speed: 50 %

**❗ Hint**

The performance of the jog TCP can be impacted by your network connection. We advise you to use an Ethernet connection to get the optimal communication and the best experience with the jog TCP feature.

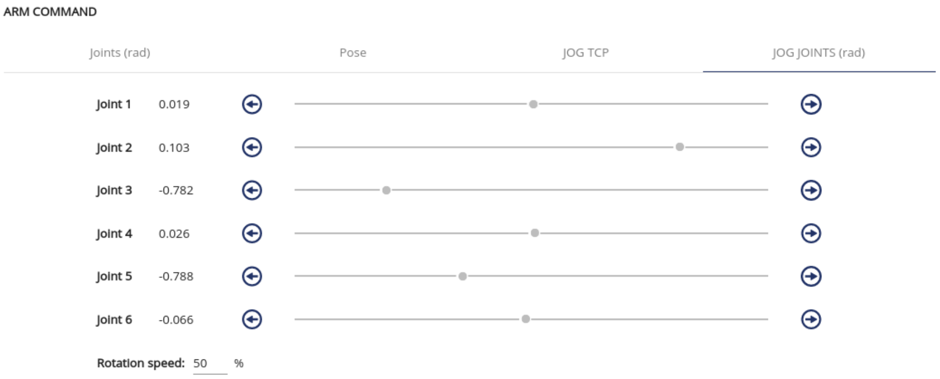
## JOG JOINTS

You can also switch to jog Joints on the ARM COMMAND panel.

The jog Joints allows you to control each joints of the robot by holding down one arrow.

The robot will move until you release the button.

You can also change the movement speed. A low value will give you more precision while a higher value will bring you faster to the position.



| | |
|---|---|
| Joint 1 | 0.019 |
| Joint 2 | 0.103 |
| Joint 3 | -0.782 |
| Joint 4 | 0.026 |
| Joint 5 | -0.788 |
| Joint 6 | -0.066 |

Rotation speed: 50 %

**❗ Hint**

The performance of the jog joints can be impacted by your network connection. We advise you to use an Ethernet connection to get the optimal communication and the best experience with the jog Joints feature.

## Tool command

- The “SCAN” button will detect and add a tool if there is already a connected tool (with motor).
- **If you have a gripper:**
    - You can set the opening and closing speed by writing the value or by dragging the cursor (from 100 and 1000),
    - You can send open and close commands to the gripper by clicking “OPEN” and “CLOSE” buttons.
- If you have a Vacuum Pump, the “ON” button will pull the air with the Vacuum Pump and the “OFF” button will pull the air out of the pump.

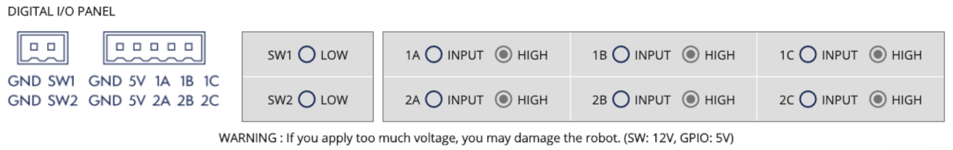
TOOL COMMAND

Current Tool: Vacuum pump

ON OFF

SCAN

## Digital I/O panel



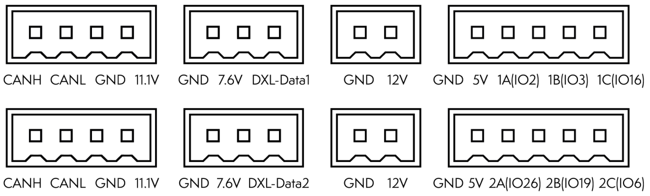
DIGITAL I/O PANEL

SW1 ○ LOW | 1A ○ INPUT ◉ HIGH | 1B ○ INPUT ◉ HIGH | 1C ○ INPUT ◉ HIGH

SW2 ○ LOW | 2A ○ INPUT ◉ HIGH | 2B ○ INPUT ◉ HIGH | 2C ○ INPUT ◉ HIGH

WARNING : If you apply too much voltage, you may damage the robot. (SW: 12V, GPIO: 5V)

You can monitor and set I/O signals from or to Ned. The “DIGITAL I/O PANEL” displays the current state of the I/O “SW1”, “SW2”, “GPIO1”, and “GPIO2” connectors on the back of Ned.



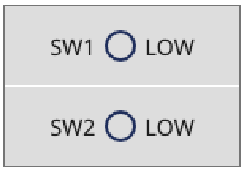
> **Note**
>
> Check the Electrical Interface (https://docs.niryo.com/product/ned/source/hardware/electrical_interface.html#electrical-interface) for electrical and connection details.

- “SW1” and “SW2” connectors can be used to plug a fan, a motor, etc.
- For “GPIO1” and “GPIO2”, you have, from left to right: GND, 5V, and 3 digital pins. The digital pins can be used to communicate with another device (ex: an Arduino board. Check out this tutorial (https://docs.niryo.com/applications/ned/source/tutorials/control_ned_arduino.html) to learn more about that).

### Switches

The switches are already set as OUTPUT mode.

You can change the “SW1” and “SW2” states to “HIGH” or “LOW” by clicking the corresponding button.



### GPIO

**Warning**

Make sure to read the Safety Precautions ( https://docs.niryo.com/product/ned/source/hardware/safety_instructions.html) and the hardware manual ( https://docs.niryo.com/product/ned/source/hardware/electrical_interface.html#digital-inputs-outputs) before using the digital I/O of Ned.

A digital pin (GPIO) can be set as an INPUT or OUTPUT mode.

You can change the mode (INPUT or OUTPUT) and the state (LOW or HIGH) for each pin, by clicking on the corresponding mode and state on the “DIGITAL I/O PANEL”.

- When the pin is set as an INPUT, you can read the state in this panel.
- In OUTPUT mode, you can set the OUTPUT state of the pin (LOW or HIGH) by clicking on the corresponding button.

**Note**

By default, all the GPIO pins are in INPUT mode and HIGH state.

#### Electromagnet control

Make sure to read the Safety Precautions ( https://docs.niryo.com/product/ned/source/hardware/safety_instructions.html) and the Hardware Manual ( https://docs.niryo.com/product/ned/source/hardware/electrical_interface.html#digital-inputs-outputs) before using the digital I/O of Ned.

The Electromagnet can be set up and controlled on the “DIGITAL I/O PANEL”.

You can connect the electromagnet on the GPIO1 or GPIO2 at the back of Ned (seeHardware Manual (https://docs.niryo.com/product/ned/source/hardware/electrical_interface.html#electromagnet)).

Then, you need to set up the digital I/O pin.

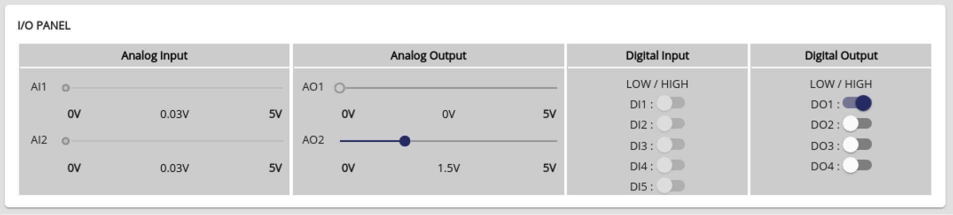
- Select the pin where you connected your electromagnet cable.
- Set this pin to OUTPUT.
- Control your electromagnet by changing the state of the selected pin to LOW or HIGH.

**Hint**

If you are using the connector provided with the electromagnet,**select 1A pin if you connect to the GPIO1** and **2A if you connect to the GPIO2**.

## I/O panel (for Ned2)

On Ned2 you will now find both digital and analog I/O. Here is how to identify them on the panel: - AO = Analog Output - AI = Analog Input - DO = Digital Output - DI = Digital Input

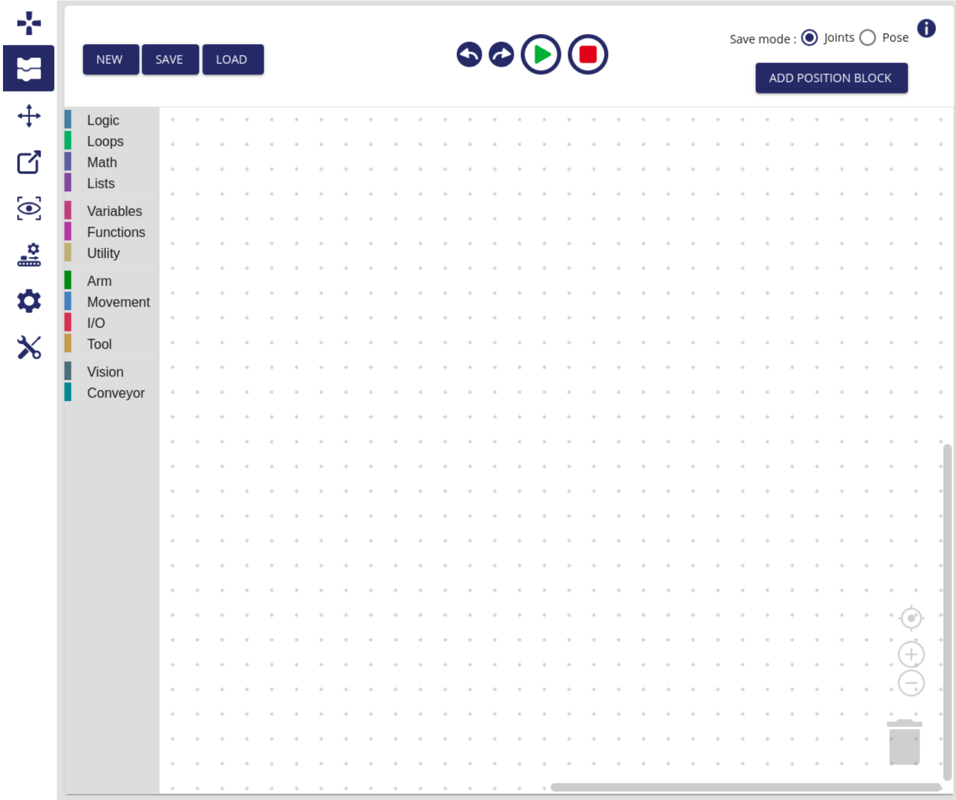


Inputs are not editable via Niryo Studio, but outputs are. The input values are directly sent back by the robot. The digital outputs can be set to “LOW” or “HIGH”. To change them, click on the toggle button. The analog outputs can be modified via the slider and can take a value between 0 and 5.

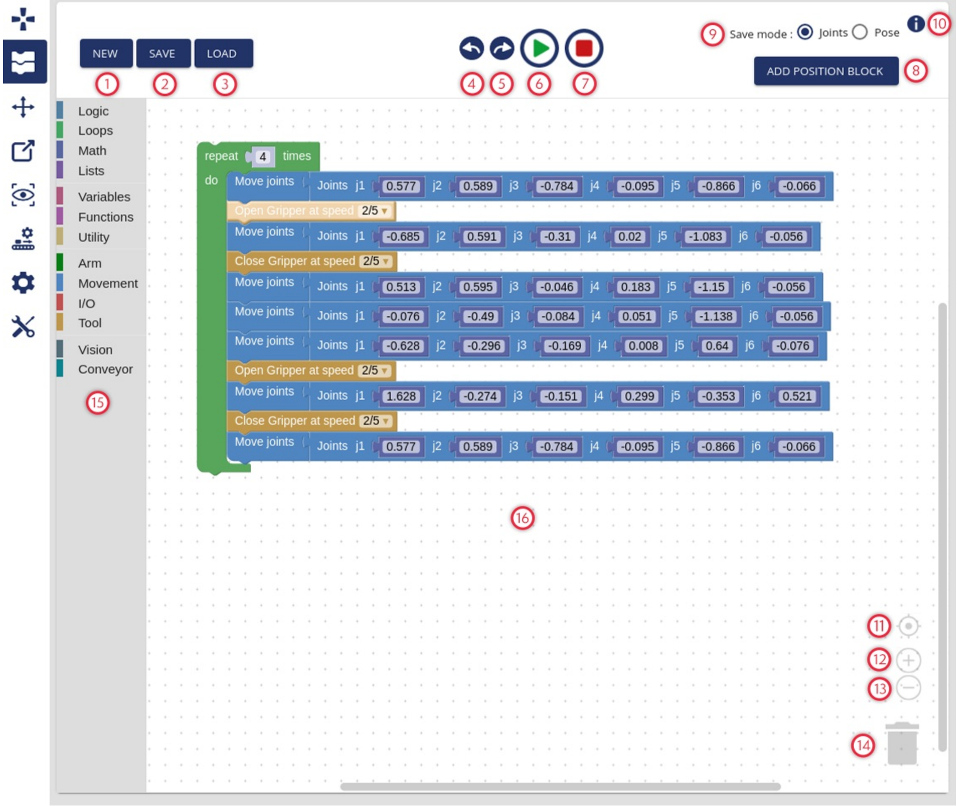
# Programming

## Project panel

Click the project button on the left menu to start creating and editing Blocky programs.

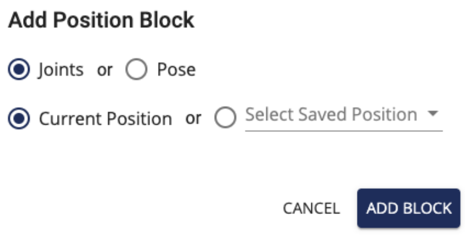

Save mode : ● Joints ○ Pose

In the figure below, the top is the project Toolbar, and the left is the Blockly menu.



```
repeat 4 times
do  Move joints  Joints j1 0.577 j2 0.589 j3 -0.784 j4 -0.095 j5 -0.866 j6 -0.066
    Open Gripper at speed 2/5
    Move joints  Joints j1 -0.685 j2 0.591 j3 -0.31 j4 0.02 j5 -1.083 j6 -0.056
    Close Gripper at speed 2/5
    Move joints  Joints j1 0.513 j2 0.595 j3 -0.046 j4 0.183 j5 -1.15 j6 -0.056
    Move joints  Joints j1 -0.076 j2 -0.49 j3 -0.084 j4 0.051 j5 -1.138 j6 -0.056
    Move joints  Joints j1 -0.628 j2 -0.296 j3 -0.169 j4 0.008 j5 0.64 j6 -0.076
    Open Gripper at speed 2/5
    Move joints  Joints j1 1.628 j2 -0.274 j3 -0.151 j4 0.299 j5 -0.353 j6 0.521
    Close Gripper at speed 2/5
    Move joints  Joints j1 0.577 j2 0.589 j3 -0.784 j4 -0.095 j5 -0.866 j6 -0.066
```

1. Create a new project: click the “NEW” button to create a new project.
2. Save program: click the “SAVE” button, to save a program. You can save a program on your computer or directly on Ned as an XML file.
3. Load project: click the “LOAD” button to import a Blockly program either from computer or from Ned. The file is a xml.
4. Undo. You can also use CTRL + Z.
5. Redo. You can also use CTRL + MAJ + Z.
6. Play the program displayed on the workspace. Once the program is done (success or not), you will get a notification on the bottom of the screen.
7. Stop the current program execution.
8. Add a position block. This will open a pop-up:



**Add Position Block**

● Joints or ○ Pose

● Current Position or ○ Select Saved Position ▾

CANCEL ADD BLOCK

This is the most useful way to add a position command block to your workspace. You can choose between a “Move Joint” block and a “Move Pose” block, and use either the current robot state, or a previously saved position.

9. Select the save mode when using the top button on Ned.

### Hint

While the Niryo Block interface is open, you can add a “Move joint” or “Move pose” block with current joints values and pose values just by pressing and releasing the top button of the robot.

10. Help.

11. Clicking on this icon will center the workspace on your blocks.
12. Workspace zoom control.
13. Workspace zoom control.
14. To delete a block, simply drag it and drop it onto the trash. You can also select it and press the delete key on your keyboard.
15. The "Niryo Robot" functions in the block library cover almost all of the possible commands you can execute on Ned:
    - Arm commands
    - Movement commands
    - Tool commands
    - I/O setup and control
    - Vision commands
    - Conveyor control

> **Note**
>
> The "NIRYO BLOCKS" panel contains everything you need to create complete programs for Ned. This is probably the panel on which you will spend most of your time.

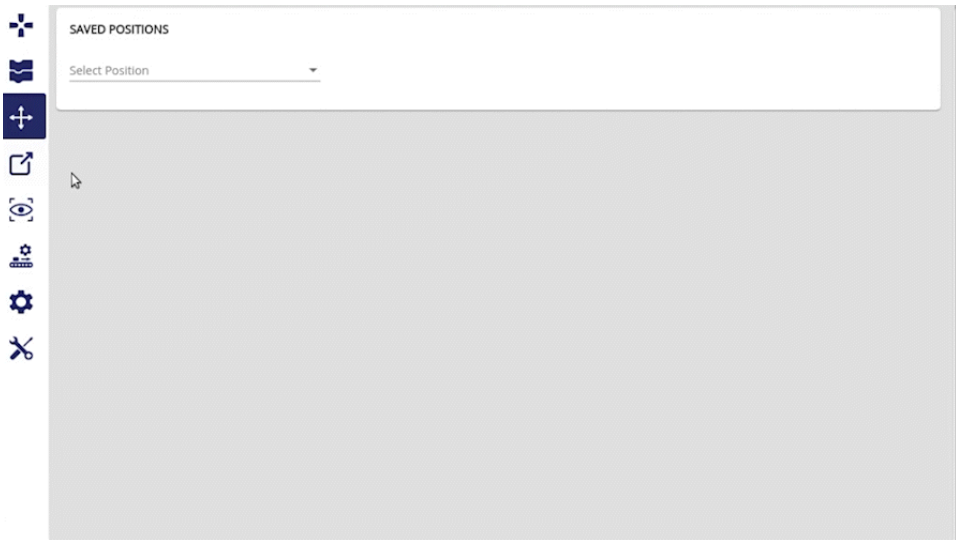
16. This is the workspace. Your whole program will be there.

> **Note**
>
> You cannot execute a new sequence while another sequence is running. If you want to start a new sequence, you will have to stop the previous one.

> **Hint**
>
> Niryo blocks along with the learning mode is the perfect combination to easily create programs in no time. Activate the learning mode, then for each position you want, add a position block corresponding to the current state, and that's it.

## Saved positions

In the "SAVED POSITION" panel, you can find all the saved positions.



When you select a position, you can see its properties:

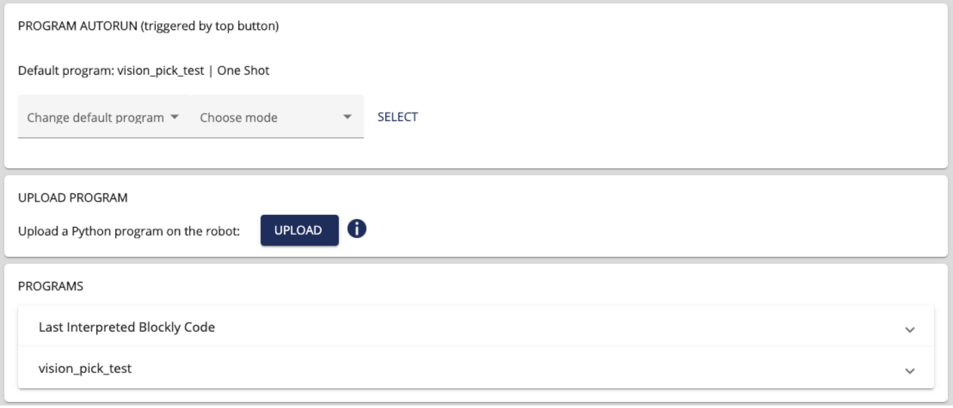
- Name
- Description
- Values: list of joints values in radian and the position and orientation of the TCP (Tool Center Point).

You can:

- Edit the position's name and description,
- Move the robot to the desired position,
- Delete the position.

## Program settings

In the "Program Settings", you can set an autorunning program, upload a Python program and display a program.



PROGRAM AUTORUN (triggered by top button)

Default program: vision_pick_test | One Shot

UPLOAD PROGRAM

Upload a Python program on the robot:

PROGRAMS

Last Interpreted Blockly Code

vision_pick_test

### Program Autorun

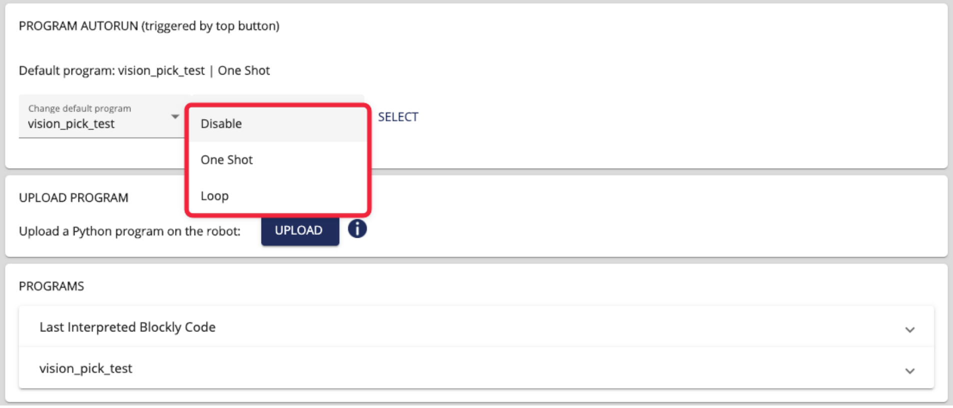
The "PROGRAM AUTORUN" functionality allows you to execute a set of sequences, directly by pressing the top button of the robot.

Here is how it works:

- Program a sequence with Blockly.
- Save the sequence on Ned.
- Add the sequence to the sequence autorun.
- Now trigger the sequence with the top button, without having to open Niryo Studio.

**Hint**

This is very useful when you use Ned in an exhibition event or for a demo. You will just need to power on Ned, wait for the LED to turn blue or green, and then press the button to run a given set of programs (that you previously programmed).



PROGRAM AUTORUN (triggered by top button)

Default program: vision_pick_test | One Shot

Change default program: vision_pick_test

Upload a Python program on the robot:

1. Display the default program triggered with the button
2. Select the program
3. Choose the “PROGRAM AUTORUN” mode. You have choices:

    a. ONE SHOT: when you press and release the top button, the given set of sequences will be run once. After that, Ned will go back to a “resting position” and activate the “Learning Mode”. You can start another run by pressing the button again. If you press the button while the set of sequences is running, it will stop the execution.
    b. LOOP: when you press and release the top button, the selected program will run in a loop, forever. You can press the button again to stop the execution.

4. SELECT button: select the program autorun.

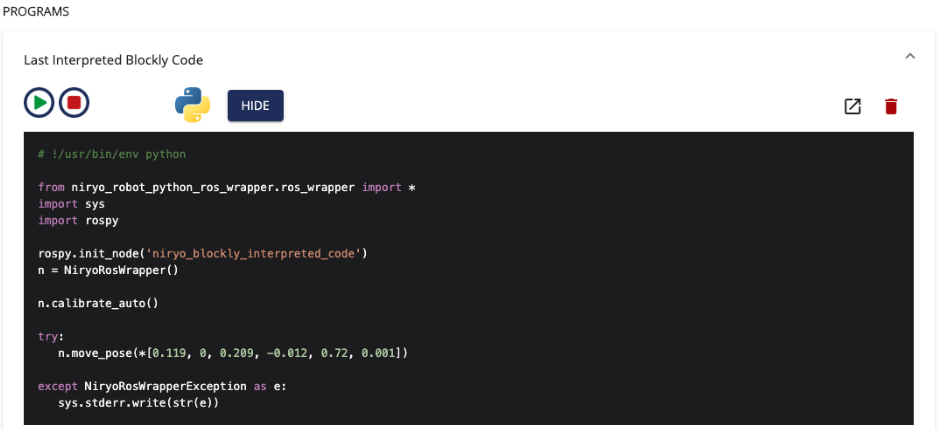
## Upload program

You can upload a Python script from your computer to Ned.

## Programs list

You can pick a previously saved program from the select box. Click on a sequence to see all the properties and actions available for this program.

You can display the Python code of the program.



```
# !/usr/bin/env python

from niryo_robot_python_ros_wrapper.ros_wrapper import *
import sys
import rospy

rospy.init_node('niryo_blockly_interpreted_code')
n = NiryoRosWrapper()

n.calibrate_auto()

try:
   n.move_pose(*[0.119, 0, 0.209, -0.012, 0.72, 0.001])

except NiryoRosWrapperException as e:
   sys.stderr.write(str(e))
```

After the program details, you get a series of available actions:

1. Play the selected program (same as the “PLAY” button in Niryo blocks).
2. Stop the current program execution (same as the “STOP” button in Niryo blocks).
3. Open the program in Niryo blocks. This will make you switch to the “Niryo blocks” panel, and the program will be added to the current workspace.

> **Hint**
>
> This functionality is very useful when you want to duplicate and create a new program from an existing one.

4. Edit the program. You can modify the name, the description, and the Blockly XML.
5. Delete the program.

# Addons

## Conveyor Belt

The use of the Conveyor Belt is detailed in its documentation ( https://docs.niryo.com/product/conveyor-belt/source/how_to_use.html#on-niryo-studio).

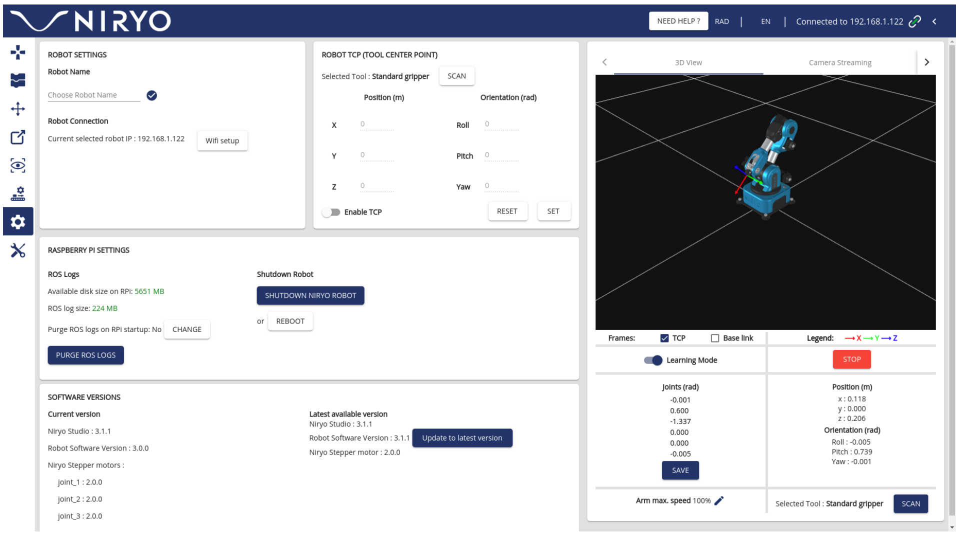
## Vision Set

Discover how to use the vision in the Vision Set documentation ( https://docs.niryo.com/product/vision-set/source/setup.html#software).

# Robot's settings and update

## Ned settings

Ned's related parameters can be set in the "ROBOT SETTINGS" panel.



ROBOT SETTINGS

Robot Name

Choose Robot Name

Robot Connection

Current selected robot IP : 192.168.1.122

ROBOT TCP (TOOL CENTER POINT)

Selected Tool : Standard gripper

| Position (m) | | Orientation (rad) | |
|---|---|---|---|
| X | 0 | Roll | 0 |
| Y | 0 | Pitch | 0 |
| Z | 0 | Yaw | 0 |

Enable TCP

RASPBERRY PI SETTINGS

ROS Logs

Available disk size on RPi: 5651 MB

ROS log size: 224 MB

Purge ROS logs on RPi startup: No

Shutdown Robot

or

SOFTWARE VERSIONS

Current version

Niryo Studio : 3.1.1

Robot Software Version : 3.0.0

Niryo Stepper motors :

joint_1 : 2.0.0

joint_2 : 2.0.0

joint_3 : 2.0.0

Latest available version

Niryo Studio : 3.1.1

Robot Software Version : 3.1.1

Niryo Stepper motor : 2.0.0

Frames: ☑ TCP ☐ Base link

Legend: → X → Y → Z

Learning Mode

Joints (rad)

-0.001
0.600
-1.337
0.000
0.000
-0.005

Position (m)

x : 0.118
y : 0.000
z : 0.206

Orientation (rad)

Roll : -0.005
Pitch : 0.739
Yaw : -0.001

Arm max. speed 100%

Selected Tool : Standard gripper

### Robot Name

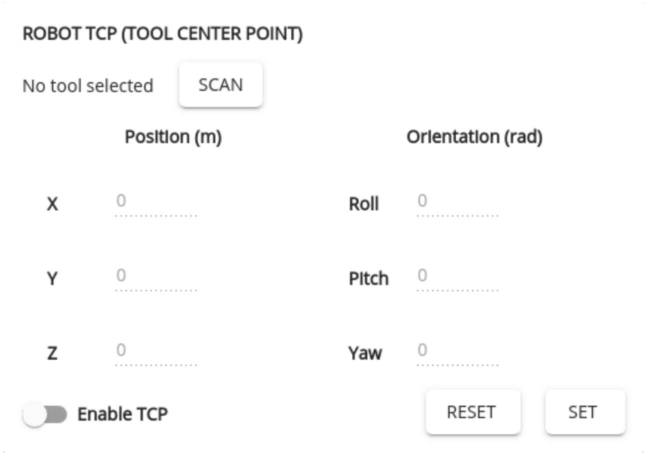
You can give a custom name to Ned.

- This name will be used when Ned is in hotspot mode to create the Wi-Fi network name.
- In Niryo Studio, when Ned is in connected mode, you will see that Ned is represented both by its IP address and its custom name.

You can set a custom name under "ROBOT SETTINGS". After choosing the name, you need to reboot Ned.

### Robot TCP (Tool Center Point)

The TCP (Tool Center Point) of the robot corresponds to the working point of the tool. From here, you can change the frame of the TCP.



ROBOT TCP (TOOL CENTER POINT)

No tool selected

| Position (m) | | Orientation (rad) | |
|---|---|---|---|
| X | 0 | Roll | 0 |
| Y | 0 | Pitch | 0 |
| Z | 0 | Yaw | 0 |

When the TCP is enabled, every position move is set corresponding to the TCP frame.

When you plug a tool and if the TCP is enabled, the TCP frame will automatically update to the corresponding tool TCP.

## Ned Wi-Fi setting

If Ned is already connected to a Wi-Fi network and you want to change your Wi-Fi network:

- Connect to Ned in “Hotspot mode” or in “Connected mode”

> **Note**
>
> If you are not connected to Ned, you will not be able to change its Wi-Fi network.

- Under “ROBOT SETTINGS”, check your current IP address.
- Click the Wi-Fi setup button and fill in the Wi-Fi name (SSID) and password of the Wi-Fi network you want Ned to connect to. Make sure to avoid typos in the name and password. Then, click on “Connect robot to Wi-Fi”.
- Ned will try to connect to the Wi-Fi, depending on the result, the robot’s reaction changes:
  - If successfully connected: the robot will reboot itself, the LED will become red then green. Connect to the same wifi network and then connect to Ned (Help: Using Ned on your Wi-Fi network (index.html#using-ned-on-your-wi-fi-network).)
  - If failed to connect to the SSID: the LED will remain blue (the LED didn’t change after ~30 seconds). Verify your password and try to connect again.
  - If the SSID can’t be found currently by the Raspberry Pi: you won’t be disconnected from Niryo Studio, but an error notification will be shown. You can retry within a few seconds (the wi-fi card may be disturbed by other wi-fi devices) or move your Raspberry Pi closer to the Wi-fi router.

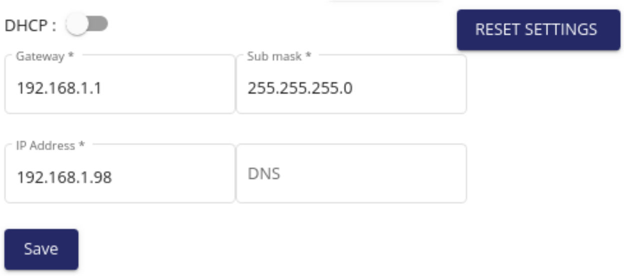
> **Warning**
>
> If you changed Ned network configuration from Wi-fi to hotspot, troubles can occur when reconnecting via Niryo Studio, please reboot your robot in this case.

## Network settings

If you are connected to Ned, you have the possibility to modify these network settings. To do so:

- Go to settings section

> **Note**
>
> If you are not connected to Ned, you will not be able to change its network settings.



DHCP :

RESET SETTINGS

Gateway * 192.168.1.1

Sub mask * 255.255.255.0

IP Address * 192.168.1.98

DNS

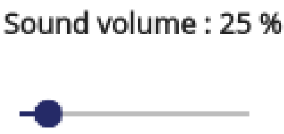
Save

- Under “Robot Connection”, you can change network settings.
- With the toggle button, you can activate or desactivate DHCP mode. This mode provides you with an automatic ip address.
- When click on “RESET SETTINGS” button, your Ned returns with the original network settings, i.e. with the ip address : 169.254.200.200
- **If you desactivate DHCP mode, you can see a form appear to modify the network settings with this information**
  - Gateway
  - Submask
  - Ip address
  - DNS
- Cclick on the “Save” button to save the information in Ned.

> **Warning**
>
> When you change the network settings, you will be automatically disconnected from Ned. You will then have to reconnect with Ned’s new IP address or search for Ned again.

### Sound settings

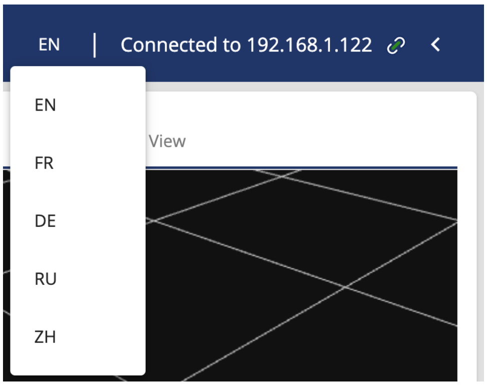
With the slide, you can adjust the robot sound’s volume from 0 to 200 percent.



Sound volume : 25 %

### Language selection

The default language of Niryo Studio is English but you can change the current language with the language button on the top toolbar.

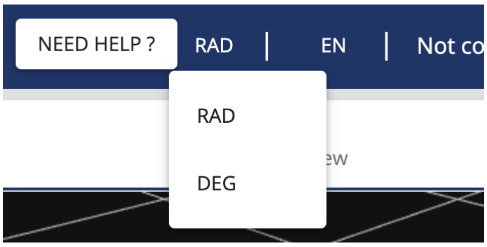


**Note**

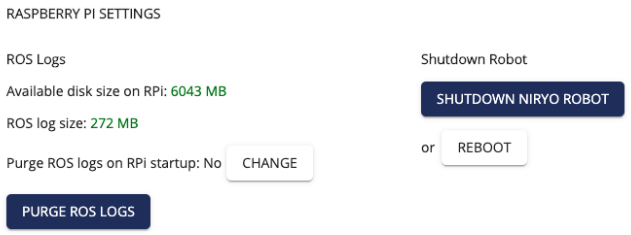
The Blockly editor and the logs remain in English.

## Angle unit selection

The default angle unit in robotics is radian. But you can change to degree with the corresponding button in the top toolbar.



## Raspberry PI settings


RASPBERRY PI SETTINGS

ROS Logs

Available disk size on RPi: 6043 MB

ROS log size: 272 MB

Purge ROS logs on RPi startup: No

Shutdown Robot

ROS logs:

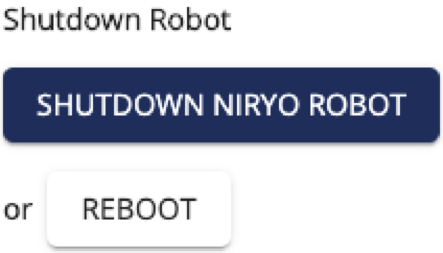
In this section, you can see:

- The available disk size on the Raspberry Pi. If the number is displayed in orange or red, it means there is no space enough on the micro-SD card.
- The size of the stored ROS logs: after some times, they can take a lot of space.
- You can choose to remove all previous ROS logs when Ned boots. This is very useful for a normal usage of Ned. Basically, you do not need all the ROS logs that are produced and stored, unless you are developing or debugging a specific point (in this case you might want to switch the option to “NO” and then turn it back to “YES” when you’re done).

**Caution**

If you never purge ROS logs (on boot or manually), the micro-SD card will eventually become full and might not work as expected. So only change the option to “NO” if you know what you are doing. In any case, if you do not really understand this functionality, always leave the option to “YES”.

- You can also remove all ROS logs immediately by clicking on the “Purge ROS logs” button.
- Shutdown Ned.



- You can shut down Ned with Niryo Studio. The “SHUTDOWN NIRYO ROBOT” will shut down the Raspberry Pi (same behaviour as if you press the physical top button for 3 seconds). Be sure to wait for the LED to turn red before completely powering off Ned.
- You can reboot Ned by clicking “Reboot”.

**Note**

If you reboot Ned, do not power it off. The LED will turn to purple, then red, then will become blue or green again.

### Software versions

You can check that Ned and relative softwares are up to date by comparing the current and latest version for each of them.

- Niryo Studio: the desktop app to control Ned.
- Robot Software: the current version of the Niryo robot ROS stack running on Ned.
- Niryo Steppers motors: a list of the firmware versions running on each NiryoStepper attached to Ned. Click here to update the firmware version (https://niryo.com/docs/niryo-one/update-your-robot/update-niryo-steppers/).

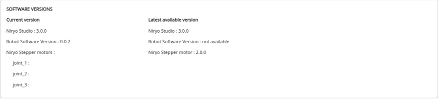
**Note**

All the NiryoStepper motors must have the same version.

- Latest available versions (directly updated from our website).

**Note**

If an incompatible software version is detected, if the version you have is not supported anymore or an update is available, you will get a pop-up notification when you connect to Ned. Please follow the displayed instructions.

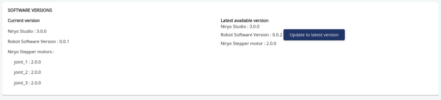


SOFTWARE VERSIONS

Current version
Niryo Studio : 3.0.0
Robot Software Version : 0.0.2
Niryo Stepper motors :
joint_1 :
joint_2 :
joint_3 :

Latest available version
Niryo Studio : 3.0.0
Robot Software Version : not available
Niryo Stepper motor : 2.0.0

a. Message displayed when Ned is not connected to internet.



SOFTWARE VERSIONS

Current version
Niryo Studio : 3.0.0
Robot Software Version : 0.0.1
Niryo Stepper motors :
joint_1 : 2.0.0
joint_2 : 2.0.0
joint_3 : 2.0.0

Latest available version
Niryo Studio : 3.0.0
Robot Software Version : 0.0.2
Niryo Stepper motor : 2.0.0

b. An update is available to update Ned software.

## Ned software Update

Ned software inside the robot can be updated through Niryo Studio in only a few steps.

First, your Ned must have an access to internet, so you have to connect it to an ethernet (you have to edit via ssh the network configuration of Ned) or a Wi-fi (index.html#using-ned-on-your-wi-fi-network) network that provides internet.

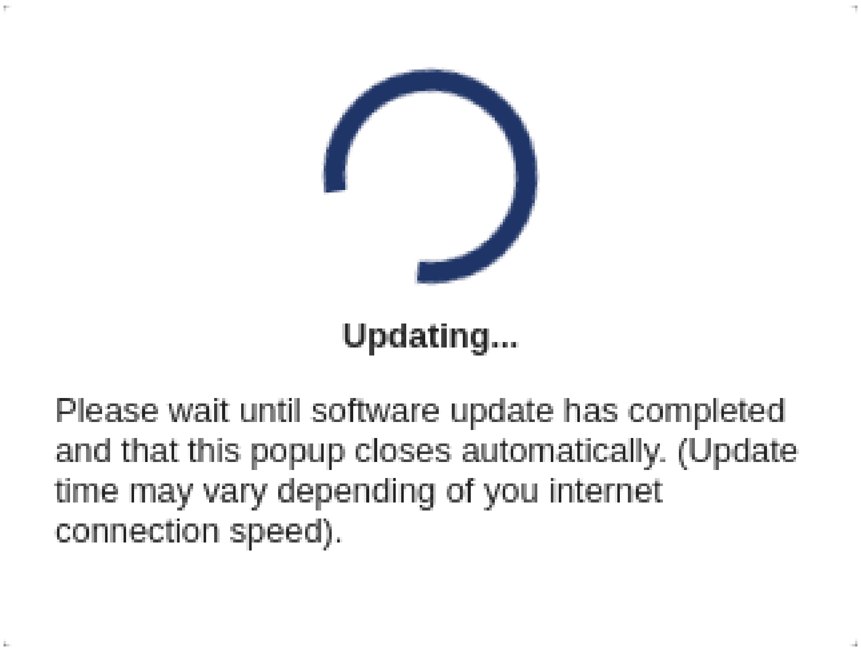
When your robot is properly configured, at the next boot of the robot, it will check if an update is available. A pop-up will then be displayed at each following connection from Niryo Studio with Ned to inform you that an update is available for your robot.

You then only have to click on the update button of the pop-up or the update button in thesoftware versions panel.

**❗ Warning**

Ned will reboot itself at the end of the process (in case of update success) so it’s recommended **not** to do any other operation with Ned before starting an update.

Once the update has started, a pop-up will indicate that Ned is updating and that it’s downloading / applying the update.



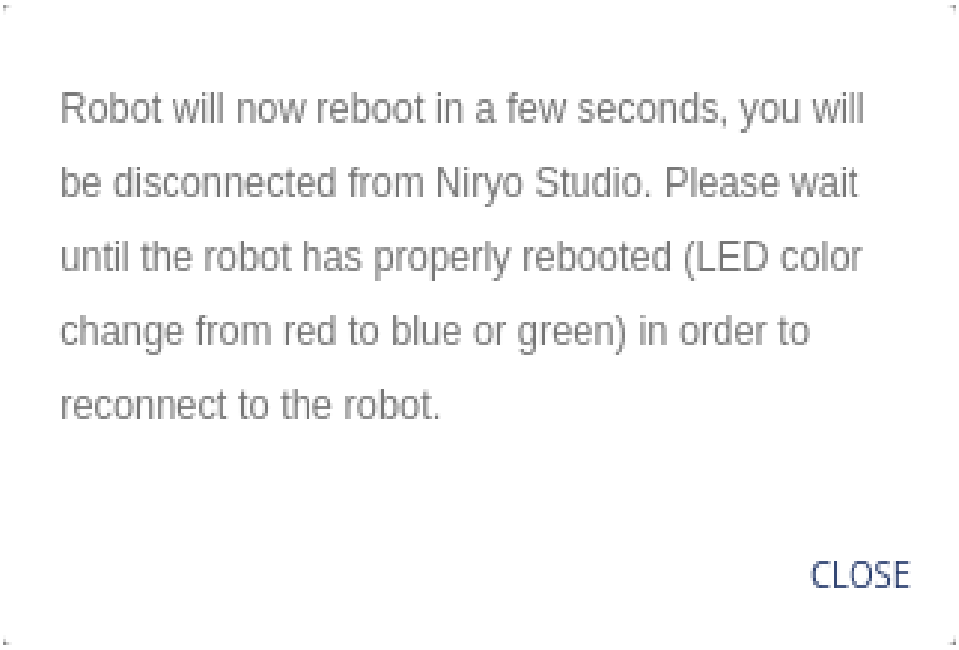
**Updating...**

Please wait until software update has completed and that this popup closes automatically. (Update time may vary depending of you internet connection speed).

**❗ Note**

Update time is heavily impacted by the internet speed of your connection so if you connect to a wi-fi network, be sure that the raspberry is not too far away from the wi-fi modem.

Once the update has successfully ended, another pop-up will appear (closing the previous one) telling that your robot will reboot and you may be disconnected from Niryo Studio. Now you just have to wait for the robot to initialize properly (the red LED turning into green or blue) before reconnecting through Niryo Studio, the update process is done.



Robot will now reboot in a few seconds, you will be disconnected from Niryo Studio. Please wait until the robot has properly rebooted (LED color change from red to blue or green) in order to reconnect to the robot.

**Warning**

In case the update failed, it could be a faulty connection, try to do the update once more and if you still encounter a problem, please contact us at contact@niryo.com (mailto:contact%40niryo.com).

# Logs and Status

## Logs

At the top of the Robot Status page, you can find 2 sections:

- Niryo Studio logs: logs produced by the desktop application.
- ROS logs: logs produced internally by Ned.

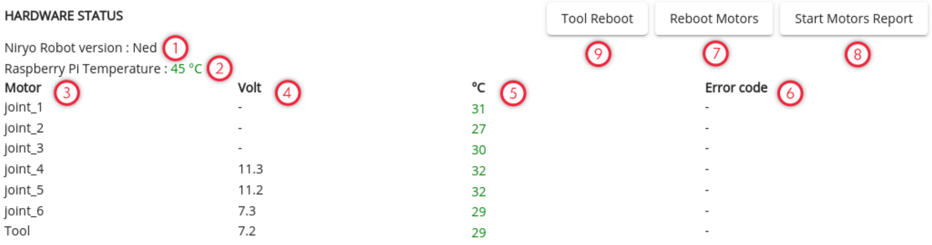
**Hint**

- You can save Ned logs or Niryo Studio logs on a text file.
- You can send this file to our technical support to help you in case of problem.

## Hardware status

In the “HARDWARE STATUS” panel, you can find the hardware status of Ned.

In this section, you can find details about the controller temperature, motors voltage, temperature, and error code.



HARDWARE STATUS

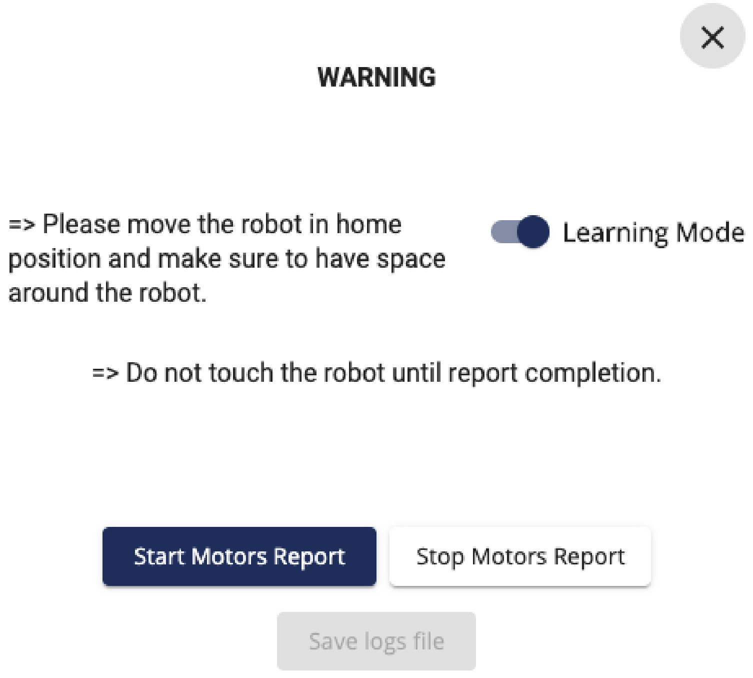
Niryo Robot version : Ned

Raspberry PI Temperature : 45 °C

| Motor | Volt | °C | Error code |
|---|---|---|---|
| joint_1 | - | 31 | - |
| joint_2 | - | 27 | - |
| joint_3 | - | 30 | - |
| joint_4 | 11.3 | 32 | - |
| joint_5 | 11.2 | 32 | - |
| joint_6 | 7.3 | 29 | - |
| Tool | 7.2 | 29 | - |

1. Ned hardware version.

2. Raspberry Pi temperature: the temperature of the Raspberry Pi is displayed in real time. If it's displayed in orange or red, that means that something is not normal and the board is too hot. Make sure that your board temperature does not exceed 70°C.
3. List of connected motors: you should have 6 motors (from Joint_1 to Joint_6), if you have a connected tool, it should appear on the list.
4. Displayed measured voltage of servo motors (Dynamixels).
5. Measured temperature of each joint.
6. Error code: displays error code if there is any. Check troubleshooting chapter to resolve the problem.
7. Reboot motors: this button reboots servo motors without rebooting the entire robot, this can help you to resolve motors errors.
8. Start motors report: this option helps you to debug Ned. Click the Motor Report button and Start Motors Report.
9. Tool reboot: this option allows you to reboot the tool's motor. This button is useful when a motor error occurs.



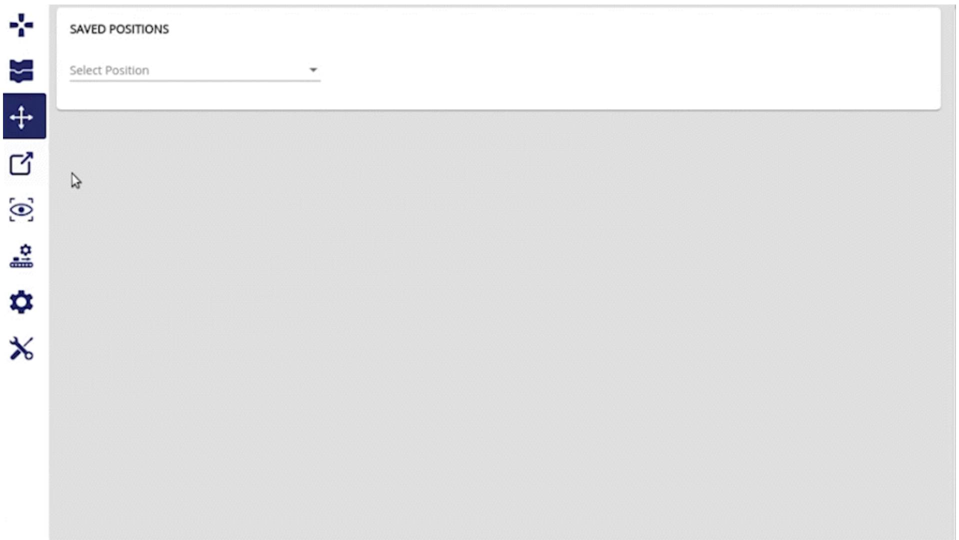
WARNING

=> Please move the robot in home position and make sure to have space around the robot.

=> Do not touch the robot until report completion.

## Sounds

In the "SAVED SOUNDS" panel, you can find all the saved sounds.

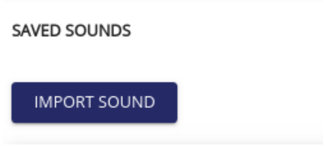


> **Warning**
>
> You must be connected to a Ned2 to see this panel.

In this panel, you can: - Import sound - Play / Stop sound - Adjust volume - Delete sound

## Import sound

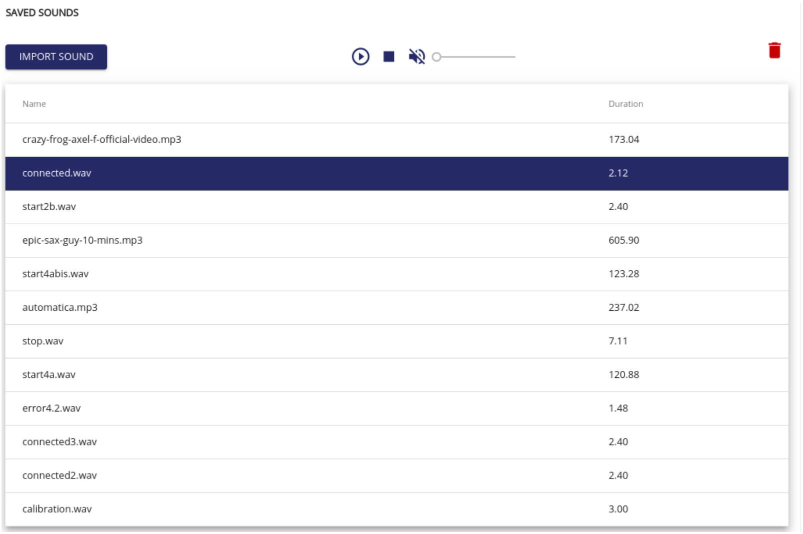
You can also import sounds to your robot. To do this, click on the “IMPORT SOUND” button, a window for selecting your file will appear.



Limitations: - Format: wav, mp3 - Size: Under 10 Mo

## Play / Stop sound

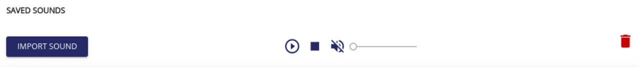
At the top center of the panel, you can find the buttons to play or stop a sound.



| Name | Duration |
|---|---|
| crazy-frog-axel-f-official-video.mp3 | 173.04 |
| connected.wav | 2.12 |
| start2b.wav | 2.40 |
| epic-sax-guy-10-mins.mp3 | 605.90 |
| start4abis.wav | 123.28 |
| automatica.mp3 | 237.02 |
| stop.wav | 7.11 |
| start4a.wav | 120.88 |
| error4.2.wav | 1.48 |
| connected3.wav | 2.40 |
| connected2.wav | 2.40 |
| calibration.wav | 3.00 |

To play or stop a sound, you must first select it by clicking a sound in the list of sounds available on the robot. When a sound is selected, the line appears in blue.

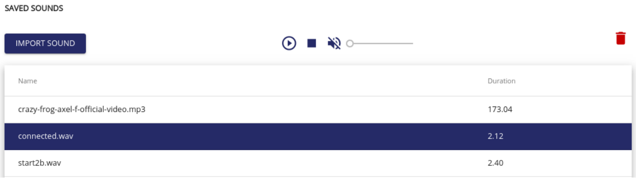
### Adjust volume



With the slide, you can adjust the robot sound volume from 0 to 200 percent.

If you click on the speaker button, you mute the robot.

### Delete sound

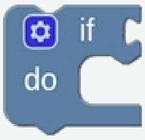
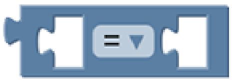
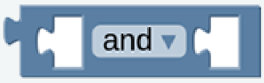
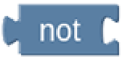
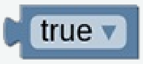
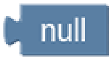
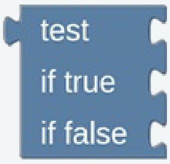
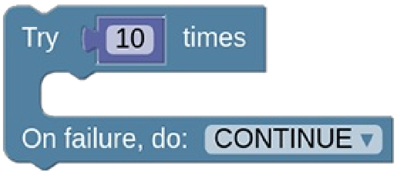


| Name | Duration |
|---|---|
| crazy-frog-axel-f-official-video.mp3 | 173.04 |
| connected.wav | 2.12 |
| start2b.wav | 2.40 |

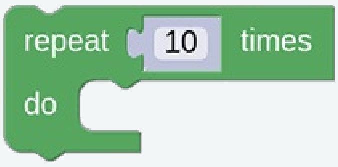
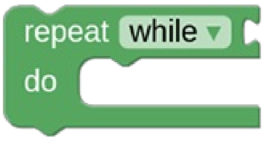
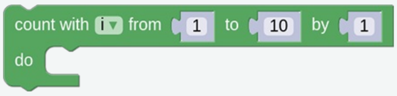
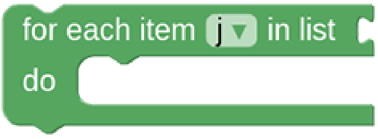
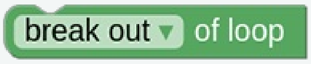
At the top of the panel, you can see trash icon, when a sound is selected, you can delete it by clicking on this icon.

A confirmation window will ask you if you really want to delete this file.

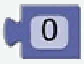
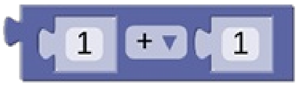
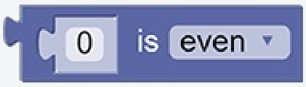
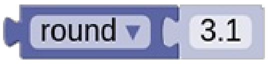
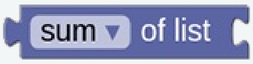
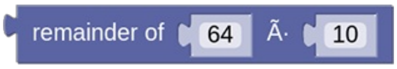
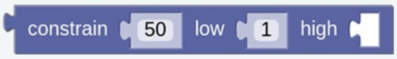
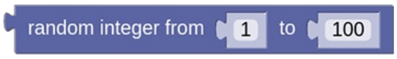
## Logic

| Concept | Explanation | Example |
|---|---|---|
| If/Else If/Else block | If the first value is true, then does the first block of statements. Otherwise, if the second value is true, does the second block of statements. If none is true, does the the last block of statements. |  |
| Mathematical comparison operator block | Compares two values with an operator [=, ≠, <, ≤, >, ≥]. Returns true if the comparison is true. |  |
| Logical comparision operator block | Compares two values with an operator [*and / or*]. Returns true if the comparison is true. |  |
| Not block | Returns true if the input is false, and false if the input is true. |  |
| True/False block | Returns either true or false. |  |
| Null block | Returns null. |  |
| Test block | Checks the condition in test block statement.<br>If the condition is true, returns the value of the ‘if true’ value.<br>Otherwise, returns the ‘if false’ value. |  |
| Try/Except block | On the “Try” statement you can ask Ned to try to execute any type of action a certain amount of time<br>You can then choose if Ned should CONTINUE or STOP in case he is “On failure”.<br>If not Ned will execute the next block. |  |

## Loops

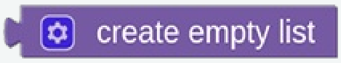
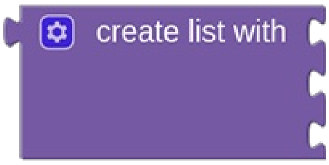
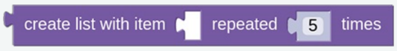
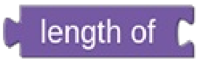
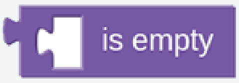
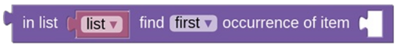
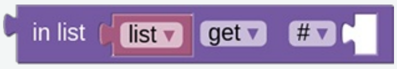
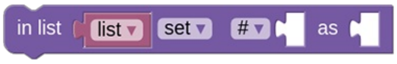
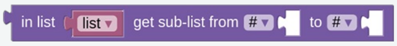
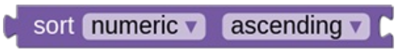
| Concept | Explanation | Example |
|---|---|---|
| Time loop block | Does one or several statements multiple times. |  |
| While loop block | While a value is true, then do some statements. |  |
| For loop block | Executes a loop based on a variable, from a starting number to an ending number with a specified interval, and executes the specified blocks. |  |
| For each item in list loop block | For each item in a list, sets the variable "v" to the item, and then does some statements. |  |
| Break block | Breaks out of the containing loop. Must be in a loop block. |  |

## Math

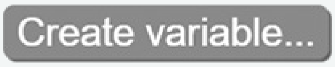
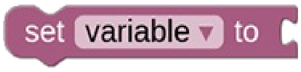
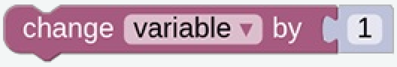
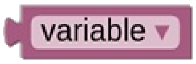
| Concept | Explanation | Example |
|---|---|---|
| Set number block | Sets a value. |  |
| Mathematical operation block | Returns the value of the specified operation with two numbers. [+, -, *x*, ÷, ^]. |  |
| Check operation block | Checks if a number is even, odd, prime, whole, positive, negative or divisible. |  |
| Round block | Rounds a number up or down. |  |
| Mathematical list operations block | Returns the mathematical operation value of all number in the list [*sum*, *min*, *max*, *average*, *median*, *modes*, *standard deviation*, *random item*]. |  |
| Division remainder block | Returns the remainder from the division of two numbers. |  |
| Constrain number block | Constrains a number to be between the specified limits (inclusive). |  |
| Random integer block | Returns a random integer between the two specified limits (inclusive). |  |

## Lists

| Concept | Explanation | Example |
|---|---|---|

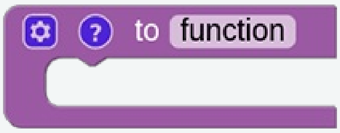
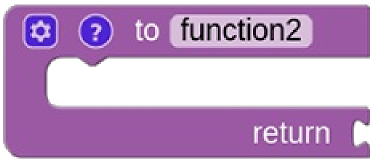
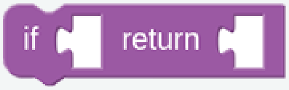
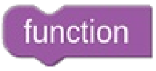
| Concept | Explanation | Example |
|---|---|---|
| Create list block | Creates an empty list. |  |
| Create list with items block | Creates a list with a specified number of items. |  |
| Create list with one item block | Creates a list consisting of the given value repeated the specified number of times. |  |
| List length block | Returns the length of a list. |  |
| Is list empty block | Returns true if the list is empty. |  |
| Item index finder block | Returns the index of the *first* or the *last* occurrence of the item in the list. Returns 0 if the item is not found. |  |
| Get item block | *Returns / returns and removes / removes* the item at the specified position in a list. #1 is the first item. |  |
| Set item index block | *Sets / inserts at* the item at the specified position in a list. #1 is the first item. |  |
| Copy list-portion block | Creates a copy of the specified portion of a list. |  |
| Sort list block | Sorts *numeric / alphabetic / alphabetic ignore case* by *ascending / descending* a copy of a list. |  |

## Variables

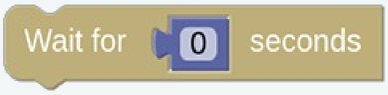
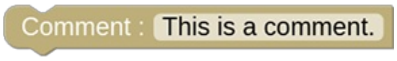
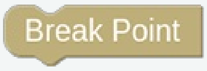
| Concept | Explanation | Example |
|---|---|---|
| Create variable block | Creates a variable by its name. |  |
| Set variable block | Sets this variable to be equal to the input. |  |
| Change variable block | Changes this variable by the input. |  |
| Get variable block | Returns the variable. |  |

## Functions

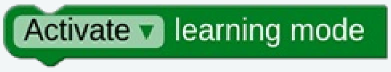
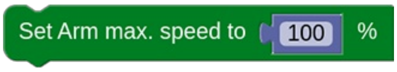
| Concept | Explanation | Example |
|---|---|---|

| Concept | Explanation | Example |
|---|---|---|
| Create no output function block | Creates a function with no ouput. You can add input arguments. |  |
| Create function with output block | Creates a function with an ouput. You can add input arguments. |  |
| Function return block | If a value is true, then ends the function and returns the return value. Must be inside a function. |  |
| Call function block | Runs the function. |  |

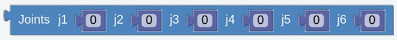
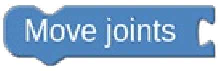
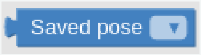
## Utility

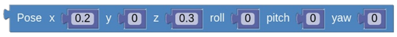
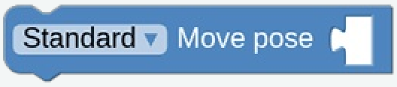
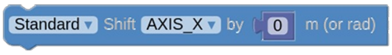
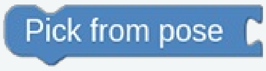
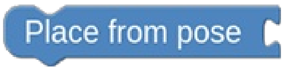
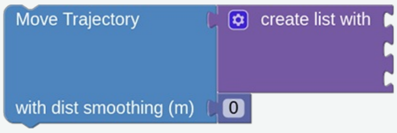
| Concept | Explanation | Example |
|---|---|---|
| Wait time block | Creates a break time in the program. |  |
| Comment block | Adds comments to the code. This block will not be executed. Note: accents are not accepted (é,à,è,...). |  |
| Break Point block | Stops the execution of the program. Press the "Play" button to resume. |  |

## Arm

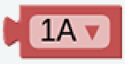
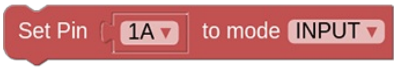
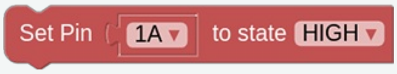
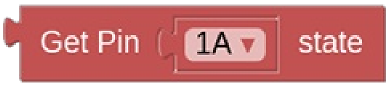
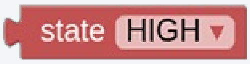
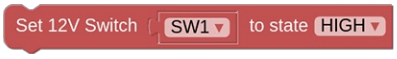
| Concept | Explanation | Example |
|---|---|---|
| Learning mode block | *Activates* / *Deactivates* the learning mode. |  |
| Set arm speed block | Sets the arm speed. |  |

## Movement

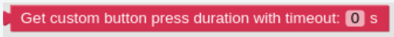
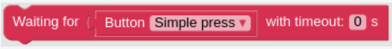
| Concept | Explanation | Example |
|---|---|---|
| Joints block | Creates an object pose according to the robot's joints values. |  |
| Move joints block | Moves the robot according to a Joints block. |  |
| Saved pose block | Creates an object pose with a pose saved in the robot. |  |

| Concept | Explanation | Example |
|---|---|---|
| Pose block | Creates an object pose according to the end effector's cartesian coordinates. |  |
| Move pose block | Moves the robot according to a pose block. You can change between "Standard" move (Point to Point), "Linear", the robot will follow a linear trajectory from its position to the desired position,<br>or "Try linear", the robot will try to do a "Linear" trajectory, but if it can't compute the linear trajectory,<br>it will do a "Standard" move. |  |
| Shift block | Shifts the robot pose according to an axe *x / y / z / roll / pitch / yaw*.<br>You can change between "Standard" move (Point to Point), "Linear", the robot will follow a linear trajectory from its position to the desired position, or "Try linear", the robot will try to do a "Linear" trajectory,<br>but if it can't compute the linear trajectory, it will do a "Standard" move. |  |
| Pick from pose block | Moves the robot's TCP (arm's end point + tool coordinate)<br>to a specified pose and activates the tool to pick an object. |  |
| Place from pose block | Moves the robot's TCP (arm's end point + tool coordinate)<br>to a specified pose and deactivates the tool to place an object. |  |
| Move Trajectory block | The robot will pass through the list of goals with the desired distance smooth.<br>The distance smooth is the radius from the goal where the robot will start to go towards the next point. |  |

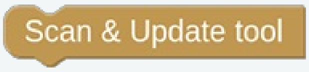
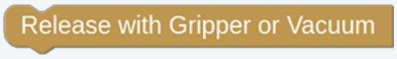
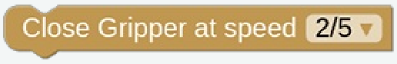
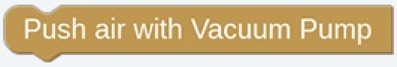
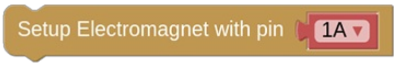
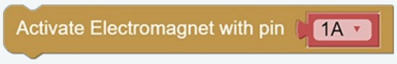
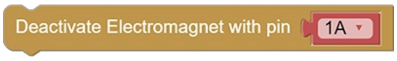
# I/O

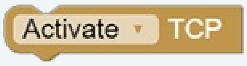
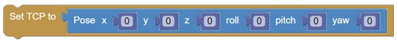
| Concept | Explanation | Example |
|---|---|---|
| Get I/O block | Returns the number of the pin. |  |
| Set I/O mode block | Sets I/O pin mode to *input* or *output*. |  |
| Set output state block | Sets output pin state to *high* or *low*. |  |
| Get input state block | Returns the input pin state. |  |
| State block | Returns the state value *high* or *low*. |  |
| Set Switch state block | Sets the 12V switch state to *high* or *low*. |  |

# I/O for Ned2

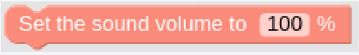
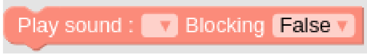
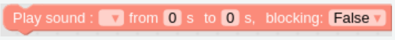
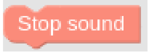
| Concept | Explanation | Example |
|---|---|---|
| Get Analog I/O block | Returns the analog pin. | AO1 ▾ |
| Get Digital I/O block | Returns the digital pin. | DO1 ▾ |
| Get Analog I/O value | Returns the value of the selected analog pin. | Get : AI1 ▾ |
| Get Digital I/O value | Returns the value of the selected digital pin. | Get : DI1 ▾ |
| Set Analog I/O value | Sets the value of the selected analog pin. | Set : AO1 ▾ to 0 |
| Set Digital I/O value | Sets the value of the selected digital pin. | Set : DO1 ▾ to LOW ▾ |
| End effector type of button press | Detects the end effector's type of button press. | Button Simple press ▾ |
| End effector button is pressed. | Detects the end effector button is pressed. | is button pressed |
| End effector custom button is pressed with timeout. | Detects the end effector custom button is pressed and sets timeout. |  Get custom button press duration with timeout: 0 s |
| End effector action button is pressed with timeout. | Detects the end effector action button is pressed and sets timeout. | Get button action with timeout: 0 s |
| End effector waiting for press button. | Waits for button press. |  Waiting for Button Simple press ▾ with timeout: 0 s |

## Tool

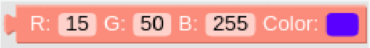
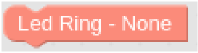
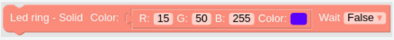
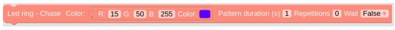
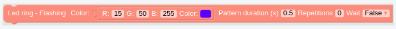
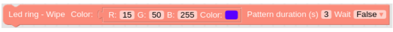
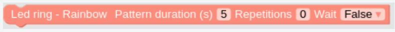
| Concept | Explanation | Example |
|---|---|---|
| Scan tool block | Scans and updates current tool. |  Scan & Update tool |
| Grasp block | Activates the *gripper / vacuum*. | Grasp with Gripper or Vacuum |
| Release block | Deactivates the *gripper / vacuum*. |  Release with Gripper or Vacuum |
| Open gripper block | Opens the gripper at a certain speed. | Open Gripper at speed 2/5 ▾ |
| Close gripper block | Closes the gripper at a certain speed. |  Close Gripper at speed 2/5 ▾ |
| Pull air block | Pulls the air in the vacuum pump. | Pull air with Vacuum Pump |
| Push air block | Pushes the air in the vacuum pump. |  Push air with Vacuum Pump |
| Setup electromagnet block | Selects on which pin the electromagnet is connected. |  Setup Electromagnet with pin 1A ▾ |
| Activate electromagnet block | Activates the electromagnet power. |  Activate Electromagnet with pin 1A ▾ |
| Deactivate electromagnet block | Deactivates the electromagnet power. |  Deactivate Electromagnet with pin 1A ▾ |

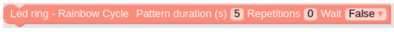
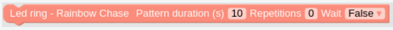
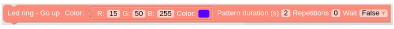
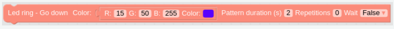
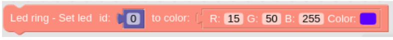
| Concept | Explanation | Example |
|---|---|---|
| Activate / Deactivate TCP block | Activates or deactives the TCP. |  Activate ▾ TCP |
| Set TCP block | Sets TCP frame |  Set TCP to Pose x 0 y 0 z 0 roll 0 pitch 0 yaw 0 |

## Sound (only for Ned2)

| Concept | Explanation | Example |
|---|---|---|
| Set sound volume | Sets the robot sound's volume. |  Set the sound volume to 100 % |
| Play sound | Plays sound selected to the robot. Possibility of blocking the execution of the program with the "Blocking" parameter. |  Play sound : ▾ Blocking False ▾ |
| Play partial sound | Plays the selected sound to the robot from X seconds to X seconds. Possibility of blocking the execution of the program with the "Blocking" parameter. |  Play sound : ▾ from 0 s to 0 s, blocking: False ▾ |
| Stop sound | Stops the current sound. |  Stop sound |

## Light (only for Ned2)

| Concept | Explanation | Example |
|---|---|---|
| Set color | Sets color with RGB parameters or color panel. |  R: 15 G: 50 B: 255 Color: ■ |
| None | No color displayed in the Led ring. |  Led Ring - None |
| Solid | Led ring solid animation, set the whole Led Ring to a fixed color. If wait is True, the blocks wait for the animation to finish. |  Led ring - Solid Color: R: 15 G: 50 B: 255 Color: ■ Wait False ▾ |
| Chase | Led ring chase animation, movie theater light style chaser animation. If wait is True and iterations not null, the blocks wait for the animation to finish. |  Led ring - Chase Color: R: 15 G: 50 B: 255 Color: ■ Pattern duration (s) 1 Repetitions 0 Wait False ▾ |
| Flashing | Led ring flashing animation, flashes a color according to a frequency. If wait is True and iterations not null, the blocks wait for the animation to finish. |  Led ring - Flashing Color: R: 15 G: 50 B: 255 Color: ■ Pattern duration (s) 0.5 Repetitions 0 Wait False ▾ |
| Wipe | Led ring wipe animation, wipes a color across the Led Ring, lights a Led at a time. If wait is True, the blocks wait for the animation to finish. |  Led ring - Wipe Color: R: 15 G: 50 B: 255 Color: ■ Pattern duration (s) 3 Wait False ▾ |
| Rainbow | Led ring rainbow animation, draws a rainbow that fades across all Leds at once. If wait is True and iterations not null, the blocks wait for the animation to finish. |  Led ring - Rainbow Pattern duration (s) 5 Repetitions 0 Wait False ▾ |

| Concept | Explanation | Example |
|---|---|---|
| Rainbow cycle | Led ring rainbow cycle animation, draws rainbow that uniformly distributes itself across all Leds.<br>If wait is True and iterations not null, the blocks wait for the animation to finish. | <br>Led ring - Rainbow Cycle Pattern duration (s) 5 Repetitions 0 Wait False |
| Rainbow chase | Led ring rainbow chase animation, like the chase animation.<br>If wait is True and iterations not null, the blocks wait for the animation to finish. | <br>Led ring - Rainbow Chase Pattern duration (s) 10 Repetitions 0 Wait False |
| Go up | Led ring go up animation, Leds turn on like a loading circle, and are then all turned off at once.<br>If wait is True and iterations not null, the blocks wait for the animation to finish. | <br>Led ring - Go up Color: R: 15 G: 50 B: 255 Color: 🟦 Pattern duration (s) 2 Repetitions 0 Wait False |
| Go down | Led ring go down animation, Leds turn on like a loading circle, and are turned off the same way.<br>If wait is True and iterations not null, the blocks wait for the animation to finish. | <br>Led ring - Go down Color: R: 15 G: 50 B: 255 Color: 🟦 Pattern duration (s) 2 Repetitions 0 Wait False |
| Set individual led | Choose the Led ID and the displayed color | <br>Led ring - Set led id: 0 to color: R: 15 G: 50 B: 255 Color: 🟦 |

## Vision

The Vision blocks and Vision templates are detailed in the Vision Set Documentation (https://docs.niryo.com/product/vision-set/source/how_to_use_blockly.html#blocks-description).

## Conveyor blocks

The Conveyor blocks are detailed in the Conveyor Belt documentation ( https://docs.niryo.com/product/conveyor-belt/source/how_to_use.html#blockly).

## Advanced Programming

There are many other ways to develop on Ned.

For more information, please refer to the following links:

- Python (https://docs.niryo.com/dev/pyniryo/index.html)
- ROS (https://docs.niryo.com/dev/ros/index.html)
- Modbus (https://docs.niryo.com/dev/modbus/index.html)

Suggest a modification

Download as PDF